SPEED-OPTIMIZED FREE-VIEWPOINT RENDERING BASED ON DEPTH LAYERING

Aleksandra Chuchvara, Olli Suominen, Mihail Georgiev, Atanas Gotchev

Tampere University of Technology, Tampere, Finland

ABSTRACT

In this paper free-viewpoint rendering is addressed and a new fast approach for virtual views synthesis from view-plus-depth 3D representation is proposed. Depth layering in disparity domain is employed in order to optimally approximate the scene geometry by a set of constant depth layers. This approximation facilitates the use of connectivity information for segment-based forward warping of the reference layer map, producing a complete virtual view layer map containing no cracks or holes. The warped layer map is used to guide the disocclusions inpainting process of the synthesized texture map. For this purpose, a speed-optimized patch-based inpainting approach is proposed. In contrast to the existing methods, patch similarity function is based on local binary patterns descriptors. Such binary representation allows for efficient processing and comparison of patches, as well as compact storage and reuse of previously calculated binary descriptors. The experimental results demonstrate real-time capability of the proposed method even for CPU-based implementation, while the quality is comparable with other view synthesis approaches.

Index Terms — free-viewpoint rendering, depth-image-based rendering, view-plus-depth, depth map, layer-based, disocclusion inpainting, local binary patterns, real time

1. INTRODUCTION

Free-viewpoint rendering (FVR) refers to a functionality allowing a user to interactively choose viewpoint and viewing direction and to freely move within a 3D visual scene. Virtual view rendering from a limited number of real views of a 3D scene has recently become a popular research area. One particular view synthesis method is depth-image-based rendering (DIBR), which generates virtual views using a single or multiple view-plus-depth inputs [1]. In this paper we consider view rendering from a single view-plus-depth input consisting of a texture image and its corresponding depth map, which associates a depth value to each image pixel. Classical DIBR techniques project points from the reference view onto a virtual viewpoints based on 3D warping equations and available depth information [2]. The main problem of DIBR is that the scene regions occluded by the foreground objects in the reference view may become visible in the virtual views. These regions contain no information and are often referred to as 'holes' or 'disocclusions'.

A number of methods have been proposed to fill in the disoccuded regions using information from available neighboring pixels. State-of-the-art disocclusion-filling methods are based on the exemplar-based inpainting algorithm proposed by Criminisi et al. [3]. The basic algorithm works in two steps and iterates until no empty pixels are left. The first step is to calculate priority for pixels along the hole border in order to determine the next patch to be filled. The priority function is a combination of a "data term" defining where the texture structure is the strongest and a "confidence term" defining the number of the known pixels in the neighborhood. The second step is to search for a patch that best matches the target patch in order to fill in the missing pixels in the target patch. This way the structural and textural information is effectively propagated in the missing regions. In the context of DIBR, disoccluded regions are assumed to contain background information. However, the original exemplar-based inpainting technique was designed for objects removal purposes, and when applied directly to disocclusion inpainting, it cannot differentiate between foreground and background areas and might propagate foreground information into the disoccluded regions. To eliminate these artifacts depth information has been utilized to guide the inpainting process [4]-[7]. In [4], a depth constraint is added to the priority function in order to favor background over foreground. A depth comparison term is also added to the similarity function, so that patches with the same depth level are considered more similar. The method proposed in [5] is based on [3] and [4]. The authors replace the data term with 3D structure tensor and calculate the priorities only along the background border to force the filling order from background to foreground. Both methods assume that the complete depth map at the virtual view is available. However, in practice this assumption does not hold. In [6] and [7], a more general case when only a warped depth map is available during the inpainting process is considered, and the holes in the warped depth map are filled simultaneously with the texture. In both approaches the source region is classified into foreground and background, and the disocclusions are filled from the background side. In [6], the priority function is modified to use more robust structure tensor together with the epipolar directional term. In [7], depth-guided directional priority is used to selects background patches. However, foreground boundaries are not always well identified by these methods due to fixed thresholding and missing depth information which lead to wrong inpainting results.

In summary, the state-of-the-art depth-aided inpainting methods consist of two stages: warped depth map inpainting and depth-aided texture map inpainting. Usually warped depth maps contain single missing pixels due to resampling. Those are considered outliers or 'cracks' and have to be removed e.g. by median filter [8]. Since depth map is a piecewise smooth grey-scale image with no high-detail texture, it is easier to be filled. The simplest approach to fill depth map is depth-based extrapolation, e.g. the disocclusion region is searched (usually in horizontal direction) and filled by minimum depth value of the found boundary pixels. In [9], the authors observe the relation between the disoccluded areas and the reference camera position. Based on this observation the filling direction is chosen to be the vector pointing from the epipole point (projection of the reference camera position onto the virtual view) to the center of the virtual view depth map. In [10], a novel forward projection technique, called Joint Projection Filling (JPF) is proposed. The virtual depth map is synthesized by the JPF method in a single step: for every projected pixel the distance between this pixel and the pixel last projected on the same row is analyzed to determine whether there is an overlap, crack or disocclusion. Then cracks

978-1-4673-8090-4/15/$31.00

are filled in by interpolation of neighboring pixels, and disocclusions are filled in by background pixels, thus the use of dedicated filtering and inpainting procedures are avoided.

In this paper a fast approach for virtual views rendering from view-plus-depth data is proposed. First the depth map is transformed into a layer map based on the disparity values associated with each depth value. This layered representation allows preserving connectivity information during the forward warping of the reference layer map. A segment-based projection approach is used to warp the layer map on a segment-by-segment basis rather than pixel-by-pixel. In contrast to classical point-based warping, the warped layer map contains no cracks or holes and there is no need to apply additional post-processing. The obtained layer map is then utilized to compute the color of the virtual view pixels and during the inpainting of the disoccluded areas in the texture map. For this purpose, a speed-optimized patch-based inpainting approach is proposed. Our approach follows the basic concepts of depth-aided inpainting methods; the layer map is used in a similar manner as depth map to guide the inpainting process, while the patch matching step is modified to use a similarity measure based on local binary descriptors. Intensity and layer data of a patch is transformed into binary descriptors, which allows for efficient processing when comparing two patches as well as compact storage of precalculated descriptors (in order to skip the calculations when the same patch needs to be evaluated as a matching pair). We illustrate the effectiveness of our approach with different view synthesis examples using synthetic and real data.

2. THE PROPOSED METHOD

In this section layer-based view synthesis method is introduced. First, the reference depth map is transformed into a layer map. Then, the forward warping of the layer map is done on a segment-by-segment basis, producing a complete virtual view layer map. The backward projection is then used to synthesize the virtual view. Finally, the inpainting is performed utilizing the computed layer map to fill disoccluded areas in texture.

2.1 Depth layering

The input is formed by the aligned texture map depth map, where the depth map reflects the distance from the reference camera image plane to the points in the scene. For a layer based system each pixel at coordinates (u,v) in the input depth map needs to be assigned to a particular layer according to its disparity value (displacement of a pixel in a virtual view). The actual depth-to-disparity mapping with respect to the targeted virtual view dependents on the distance between the cameras and their intrinsic parameters and can be calculated as follows:

$$d(u,v) = \frac{b \cdot f}{z(u,v)}, \quad (1)$$

where f is the reference camera focal length measured in pixels, b is the distance between the cameras (baseline) in meters, z is the depth value in meters and d is the corresponding disparity in pixels. The values in disparity domain are limited by the maximum and minimum disparity in the scene d_{min} and d_{max} and can be segmented into N constant disparity layers as follows: assigning pixels with disparity within the interval $[d_{min}, d_{min}+1)$ to the first layer $l_1 = d_{min}$, within the interval $[d_{min}+1, d_{min}+2)$ to the second layer $l_2 = d_{min}+1$, and so on until $d_{min}+N \geq d_{max}$. The interval $[d_{min}+N-1, d_{max})$ corresponds to the last layer $l_N = d_{min}+N-1$.

Uniformly placed layers in disparity domain become non-uniform in depth domain: layers are sparser at far depths and their density is increasing at closer depths (Figure 1). Thus, a

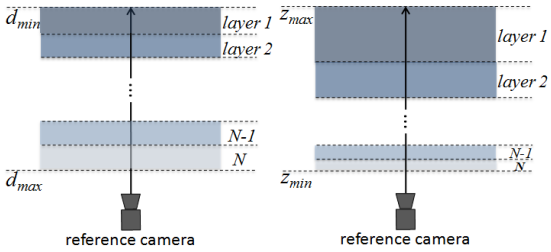

Figure 1. Depth layering in a) disparity and b) depth domain;

layer map can be seen as non-uniformly quantized depth map. The difference between the disparity values assigned to the same layer is less than one pixel, meaning that the quantization error of the layered-depth representation is less than one pixel as well. Such flat layered representation provides a good approximation of the scene geometry by a set of constant depth layers which still allows for accurate view synthesis. It also provides a way to project the reference layer map onto a virtual viewpoint preserving the connectivity information as described in the next section.

2.2 Forward warping of the layer map

DIBR techniques project points from the reference view onto a virtual viewpoints based on warping equations and available depth information. However, straightforward point-based projection results in visual artifacts, like disocclusions and cracks. In order to avoid these artifacts, we propose to use connectivity information, which can be extracted from the layered-depth representation of the scene geometry. A segment-based projection approach is used to warp the reference layer map on a segment-by-segment rather than pixel-by-pixel basis.

Consider a single line of pixels from the reference layer map. A *segment* is one or more pixels in the line belonging to the same layer. Segmentation of a line is done in such a way that a segment belonging to a lower layer (further from the camera) is not interrupted in case it is partly occluded by a segment of a higher layer (closer to the camera) as illustrated in Figure 2. This is achieved by maintaining a stack of segments. The segment structure contains two fields: layer number and start point of a segment. Let L_t is the layer number of the topmost segment in the stack, p is current pixel in the line, L_p is the layer number of the current pixel. The stack is maintained by the following simple rules:

1. a lower segment cannot be put on top of a higher segment;
2. if $L_p = L_t$: do nothing, proceed to the next pixel p+1;
3. if $L_p > L_t$: a higher segment in the line is found, place new segment on top of the stack with the layer number equals L_p and the start point equals p, proceed to the next pixel p+1;
4. if $L_p < L_t$: a lower segment in the line is found, it cannot be placed in the stack, because the 1st rule is violated. In order to place the current segment, all the higher segments should be removed from the stack, i.e. projected onto the virtual view with the end point equals p. Then the 2nd or the 3rd rule is applied;
5. if the end of line is reached, the stack should be emptied, i.e. the rest of segments in the stack should be projected onto the virtual view with the end point equals to the end of line.

Each segment has constant depth value and covers an elongated rectangular area, because every pixel covers a pixel size area, e.g. a pixel at coordinates (u,v) covers area from $(u$-0.5, v-0.5) to $(u$+0.5, v+0.5); a segment starting at (u,v) and containing three pixels covers area from $(u$-0.5, v-0.5) to $(u$+3.5, v+0.5). In order to render virtual layer map, the rectangular area corresponding to each segment have to be warped onto the virtual viewpoint: the corners of the segment are projected onto the virtual view and integer coordinates within the projected rectangle are assigned to the corresponding layer. During warping,

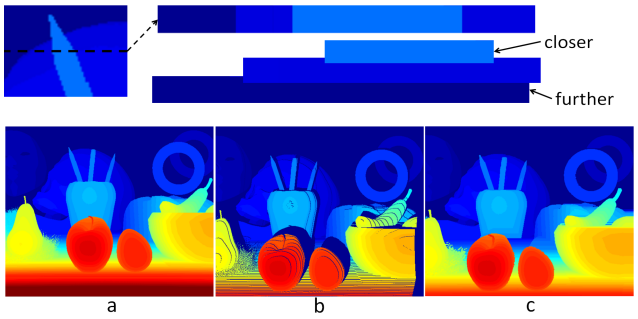

Figure 2. *Top*: line segmentation; *Bottom*: layer map warping a) initial layer map, b) point-based warping and c) segment-based warping

overlapping might appear, so it is necessary to maintain z-ordering of the layers, i.e. when projecting a segment, the layer number should not be copied to a pixel if it already contains a higher layer number.

Layering, segmentation and projection are done in a single pass through the depth map. Figure 2 presents a synthesized layer map produced by the segment-base warping. As can be seen, the synthesized layer map contains no cracks and lower layers cover disocclusions.

2.3 Virtual view synthesis

The texture of the virtual view is obtained by backward warping utilizing the synthesized layer map. The layer map can be transformed into a depth map by substituting the layer number with the corresponding depth level value and adding the *z*-component of the translation vector. Each pixel of the virtual view is warped back onto the reference view where the color of the pixel is interpolated. However, the disoccluded regions of the virtual view are warped to the same regions of the reference view as the occluding objects, thus in order to detect disocclusions the neighborhood of the projected points is checked for layer consistency. The four-point nearest neighborhood is considered. The color of the projected point can be interpolated only if the neighborhood of the projected point contains pixels belonging to the same or neighboring layers ($l\pm1$). Disoccluded regions are determined by the pixels where no value was assigned, see Figure 3.

3. INPAINTING OF DISOCCLUSIONS

After projection, disocclusion areas in the virtual texture map should be filled. For this purpose a speed-optimized patch-based inpainting approach is proposed. Our approach follows the basic concepts of the exemplar-based inpainting methods reviewed in the introduction section. It is an iterative process, which consists of two main steps: priority calculation and patch matching. In this section, we describe our proposed modifications in detail.

3.1 Layer-based priority computation

Disocclusions are assumed to contain background textures. To achieve better inpainting results, the priority function should be constructed such that disocclusions are inpainted from the background to the foreground. Priority of a pixel at coordinates (u,v) is defined as multiplication of the confidence and data terms:

$$P(u,v)=C(u,v)\cdot D(u,v), \tag{2}$$

where $C(u,v)$ and $D(u,v)$ are defined according to Criminisi's algorithm [3]. Originally, at initialization the confidence term is set to $C(u,v) = 0$ for missing pixels and to $C(u,v) = 1$ for filled pixels, thus it gives higher priority to patches which have more of their pixels already filled. In order to impose the desired filling order from the background to foreground we have modified the confidence term initialization of the filled pixels and utilized the available layer information as follows:

$$C(u,v)=\left(\frac{N-l(u,v)}{N}\right)^4, \tag{3}$$

where $l(u,v)$ is the layer value of a pixel and N is the total number of layers. This way initial $C(u,v)$ is defined over [0;1] and decreasing from the background to the foreground assigning low confidence values to the foreground pixels.

3.2 Patch matching with modified local binary patterns

Traditionally, patch similarity comparisons has been done in spatial domain using pixel differences, e.g. sum of absolute differences (SAD). In order to reduce the computational cost, the patch matching function is modified to use a similarity measure based on local binary patterns (LBP) descriptors. The LBP texture descriptors combine texture and structure properties and due to their discriminative power and computational simplicity has been widely used in various computer vision applications [11].
In its simplest version LBP descriptor is calculated by comparing the pixels in a local 3×3 window to the center pixel. To calculate patch similarity we use a modified version of LBP which compares all the pixels (including central pixel) to the mean value of the pixels in a patch. The modified LBP descriptor of a patch is defined as follows:

$$LBP_{\Omega}^{p}=\begin{cases}1, & if\ p\geq m_{\Omega}\\ 0, & if\ p<m_{\Omega}\end{cases},\quad p\in\Omega \tag{4}$$

where Ω is the patch area, p is a pixel in the patch, m_{Ω} is the mean value of the pixels in the patch.

We encode the binary LBP descriptor by bit string. The encoded patch of n×n pixels therefore has the size of n×n bits, one bit per pixel. Such binary representation of the patch allows for both compact storage of calculated descriptors, and efficient processing when comparing two patches. It makes sense to store the previously calculated descriptors in order to skip the calculations when the same patch needs to be evaluated multiple times as a candidate patch.

The layer map is considered in the patch matching process along with the texture map to find the similar patches. For a target patch containing some empty pixels, LBP descriptors are calculated using available data, i.e. the mean value is calculated over the filled pixels and only the bits corresponding to the filled pixels are taken into account when comparing two descriptors. The best matching patch is found by minimizing the following cost function:

$$\begin{aligned}cst\ (p,q)=&d(LBP_I^p(p),LBP_I^q(q))+d(LBP_L^p(p),LBP_L^q(q))+\\&+\alpha\cdot\left|m_I(p)-m_I(q)\right|+\beta\cdot\left|m_L(p)-m_L(q)\right|,\end{aligned} \tag{5}$$

where p is a target patch and q is a candidate patch in the search window (only a patch containing no empty pixels can be used as candidate patch); $LBP_I(p)$ and $LBP_I(p)$ are corresponding texture descriptors; $LBP_L(p)$ and $LBP_L(p)$ are corresponding layer descriptors; $d(\cdot,\cdot)$ is Hamming distance between the two descriptors; $m_I(p)$ and $m_I(q)$ are mean intensity values (*RGB* values are converted into intensity); $m_L(p)$ and $m_L(q)$ are mean layer values; α and β are positive weighting parameters.

The Hamming distance between two descriptors can be computed by taking XOR of the two bit strings and multiplying it by the mask bit string (AND operation) to exclude bits corresponding to the empty pixels, then counting set bits from the result (i.e. population count). A straightforward solution for counting set bits is to loop through a bit string and count one by one whether or not a bit is set. A more optimal solution that can be used on most platforms is a precalculated lookup table: the XORed bit string is interpreted as an integer, and is used to in-

dex the lookup table. For instance, using 16 bit segments, bits of a 64-bits bitstring is processed with 4 memory lookups.

Theoretically there are no constraints for the patch size. However from a practical point of view, 8×8 patch size is preferred as it results in a 64 bit representation of a patch, which efficiently maps into one or two registers in modern CPUs.

4. EXPERIMENTAL RESULTS

In our experiments we used two test video sequence, namly "Ballet" and "Break dancers". These are 8-camera sequences with baseline 300 mm and 400 mm respectively and the same spatial resolution of 1024×768 pixels [12]. Each central view 5 is used to synthesize view 4. The visual results are shown in Figure 3. The disoccluded areas are marked black. It can be seen, that our algorithm preserves objects boundaries and correctly propagates the required background information into the disoccluded areas. To evaluate the objective performance of the algorithm, peak signal to noise ratio (PSNR) evaluation metrics is used. For comparison, we have used the results presented in [6], where a table of average PSNR values over 100 frames for the methods proposed in [3][4][5] is provided. The same results along with the PSNR evaluation of the proposed method are given in Table 1. The objective evaluation results appear to be comparable to the existing algorithms, but for much lower computational cost.

Table1. PSNR evaluation of the proposed and other methods

Test seq.	**[3]**	**[4]**	**[5]**	**ours**
BA 5→4	27.09	28.91	28.70	**29.05**
BR 5→4	29.46	24.71	28.59	**28.18**

To measure the performance, the proposed method was implemented in C++ and tested on a computer with Intel Core i7-4500 1.80GHz CPU. The performance results are measured for the "Ballet" sequence: the 5th camera first frame is used to synthesize a view from the 4th camera. Figure 4 compares the execution time of our algorithm implemented with LBP-based and with SAD-based similarity measure. As expected, due to reuse of precalculated descriptors the execution time for LBP-based approach grows much slower depending on the search window size, and is still at interactive frame rate even when the search window radius is set to 50 pixels, while SAD-based implementation runs 10 times slower at this point.

5. CONCLUSION

We have proposed a speed-optimized rendering approach for virtual view synthesis from view-plus-depth data. The speed gains are achieved by employing depth map layering in disparity domain and by utilizing local binary patterns descriptors as a similarity measure for the exemplar-based texture synthesis. The layered representation facilitates a segment-based layer map warping, producing a warped layer map which contains no cracks or disocclusions, eliminating the need of additional post-processing. The binary representation of a patch data allows for efficient processing and comparison of patches, as well as compact storage and reuse of previously calculated binary descriptors. We use 8×8 patch size as it utilizes hardware efficiently, however it may be tricky to change the window size freely and retain the computational benefits.

The experimental results demonstrate the real-time capability of the proposed method even for CPU-based implementation, while the quality is comparable with other view synthesis approaches. Future work will address further improvements of the speed and quality of LBP-based inpainting and will target better temporal consistency of the inpainting results.

Figure 3. Results for proposed LBP-based inpainting. Left: synthesized view with disocclusions. Right: inpainted view.

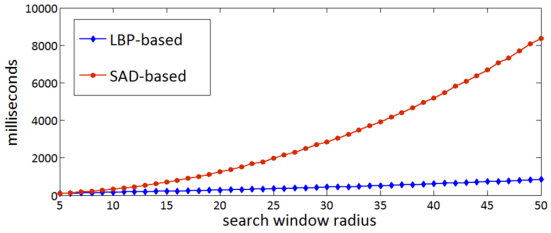

Figure 4. Execution time LBP-based and SAD-based implementation

6. REFERENCES

[1] C. Fehn, "Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV," *SPIE Stereoscopic Displays and Virtual Reality Systems XI*, pp. 93–104, Jan. 2004.

[2] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[3] A. Criminisi, P. Pérez, K. Toyama, "Object Removal by Exemplar-Based Inpainting", *IEEE Trans. Image Processing*, vol. 13, No.9, pp. 1200–1212, Sep. 2004.

[4] I. Daribo and B. Pesquet-Popescu, "Depth-aided image inpainting for Novel View Synthesis", in *MMSP,* pp. 167–170, Dec. 2010.

[5] J. Gautier, O. L. Meur, C. Guillemot, "Depth-based image completion for view synthesis," in *3DTV Conference*, 2011, pp. 1–4.

[6] I. Ahn and C. Kim, "Depth-based disocclusion filling for virtual view synthesis," in *ICME*, 2012, pp. 109–114.

[7] S. Muddala, R. Olsson, M. Sjostrom, "Disocclusion Handling Using Depth-Based Inpainting", *MMEDIA*, 2013, pp. 136–141.

[8] L. Do, S. Zinger, Y. Morvan, P. With, "Quality improving techniques in DIBR for free-viewpoint video," *3DTV Conference*, 2009.

[9] Q. Nguyen, M. Do, S. Patel, "Depth image-based rendering from multiple cameras with 3D propagation algorithm," in *IMMERSCOM*, 2009, pp. 1–6.

[10] V. Jantet, C. Guillemot, L. Morin, "Joint projection filling method for occlusion handling in depth-image-based rendering," in *3D Research*, vol.2, pp. 1-13, 2011.

[11] M. Pietikainen, A. Hadid, G. Zhao, T. Ahonen, *Computer Vision Using Local Binary Patterns*. Springer, Berlin, Germany, 2011.

[12] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, R. Szeliski, "High-quality video view interpolation using a layered representation," *ACM SIGGRAPH*, vol. 23, no. 3, 2004, pp. 600–608.